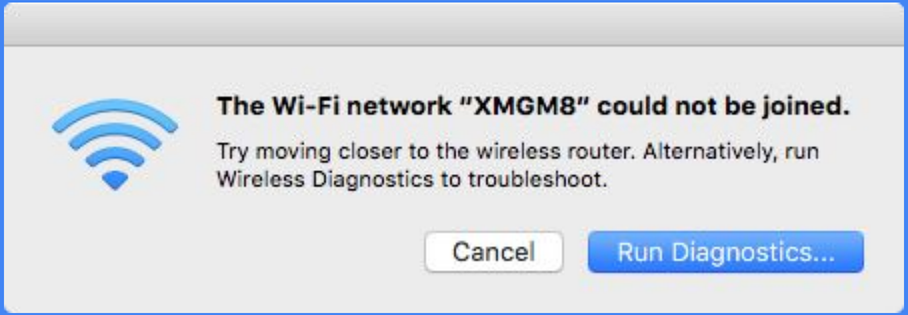
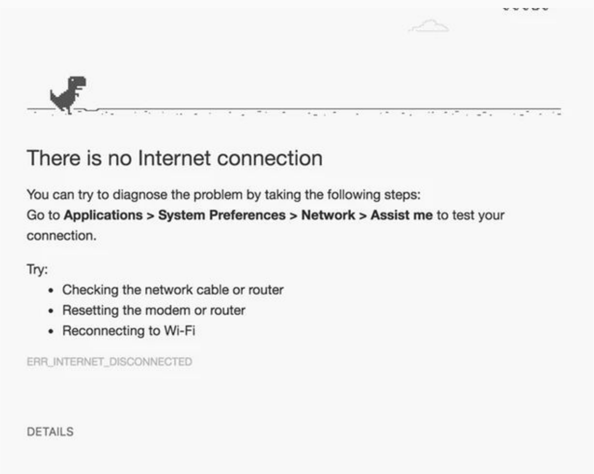
Networking 102

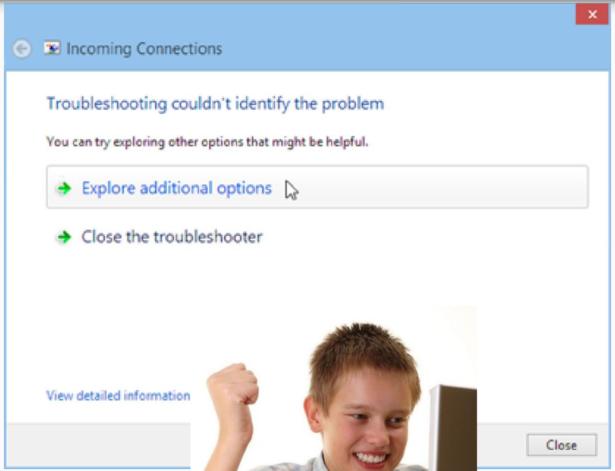
Linux SysAdmin DeCal Fall 2020
~abizer

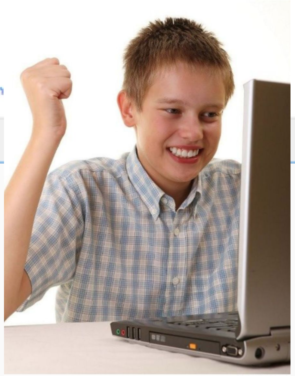
Overview



- Models of Networks
- Addressing
- Configuration Files
- Network protocols
- Sysadmin Tools







OSI Model

or CS 168 in 10 minutes

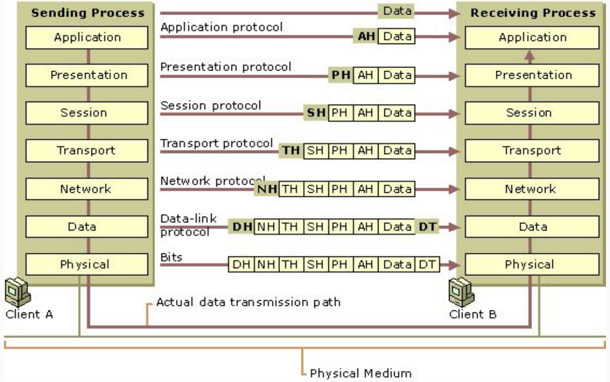
7-Layer Conceptual Model of Network Architecture

Conceptual Model: OSI

- Layers of Abstraction
  - Layers communicate with corresponding layer across hosts
  - Upper layers abstract lower layers, vice versa

- Conceptual model, doesn't always map to reality
  - Other models (e.g. TCP/IP) exist

- Packet Path
  - Sending : top to bottom, app. -> physical
  - Receiving: bottom to top, phys. -> application



OSI - Media Layers

- Layer 1: physical media
  - communication: one physical port to another
  - copper cables, lasers, radio waves, e.g. WiFi 802.11(a,b,g,n,ac)
  - line codes (e.g. 8b/10b, 64b/66b) for clock recovery

- Layer 2: data link layer
  - communication: one NIC to another
  - low-level hardware interface to physical media
  - MAC address “hardcoded” into device
  - local routing between physically-connected nodes
  - ARP, NDP

- Layer 3 - Network Layer

OSI - Host Layers

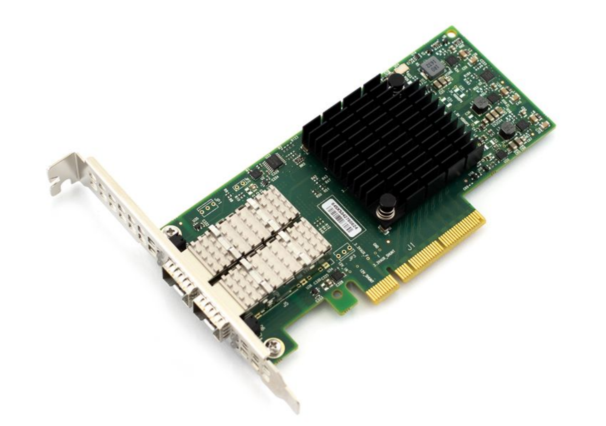
- Layer 4 - Transport Layer
  - connection: one service (on a host) to another
  - options: reliable, connection-oriented transport (TCP), or connection-less (UDP)
  - multiplexes several services into a single logical address
- Layer 5 - Session Layer
  - not commonly used outside of RPC
  - established common session information on top of reliable transport
- Layer 6 - Presentation Layer
  - barely relevant anymore
- Layer 7 - Application Layer

Layers and tools available at each

Layer 2: link layer network interface

- Actual hardware* that lets you connect to a network
- Port and cable (ethernet) or antenna (wifi)
- may correspond to physical NICs or virtual devices, e.g. loopback interface, virtual bridge, etc.
- associated with MAC address
- On Linux, interfaces can be listed through ip link
    - eno1, ens5s0f1d1, wlp9s0…
    - systemd “predictable” interface naming

* or virtual



Network Interface Card or NIC

ip link

1. lo: loopback interface, packets return to my machine
2. wlp9s0: wifi antenna
3. enp10s0: motherboard ethernet
4. zt0/zt1: zerotier virtual private network interface devices

MAC address

```
abizer@hadar ~
❯ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: wlp9s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq s
    link/ether a0:af:bd:c0:89:ab brd ff:ff:ff:ff:ff:ff
3: enp10s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc m
    link/ether 70:85:c2:4e:aa:62 brd ff:ff:ff:ff:ff:ff
4: zt1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 2800 qdisc fq_code
    link/ether 0e:13:14:01:7e:c9 brd ff:ff:ff:ff:ff:ff
5: zt0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 2800 qdisc fq_code
    link/ether e6:a0:8a:86:b8:1a brd ff:ff:ff:ff:ff:ff
```

maximum transmission unit size

UP: logical interface is enabled in kernel
LOWER_UP: physical interface on, cable plugged in, L2 works

MAC Addresses and ARP

- Media Access Control
  - hardware-level address that lets multiple physically-connected nodes address one another
  - 48 bits, 6 octets - 00:14:22:01:23:45 / broadcast address: ff:ff:ff:ff:ff:ff
  - first 3 octets are organization unique identifier (hw mfg)
- ARP: Address Resolution Protocol
  - converts L3 (logical) address to L2 (link) address
  - allows interface between global routing and local routing
  - kernel caches 192.168.1.1 -> 1f:32:af:01:65:db, default 60s
  - on receiving a request to forward a packet to logical address with unknown phys addr, broadcast an ARP request on appropriate interface for next hop based on routing table

ARP subsystem

$ arp - look at entries in the system's ARP table (or $ ip neigh)

```
abizer@surge:~$ arp -e
Address                  HWtype  HWaddress           Flags Mask            Iface
pileup.OCF.Berkeley.EDU          (incomplete)                              eno1
death.OCF.Berkeley.EDU           (incomplete)                              eno1
dementors.OCF.Berkeley.  ether   52:54:00:a6:16:f6   C                     eno1
monsoon.OCF.Berkeley.ED  ether   52:54:00:f4:1c:cc   C                     eno1
dhcp-169-229-226-145.OC  ether   4c:80:93:d3:5c:4d   C                     eno1
whirlwind.OCF.Berkeley.  ether   52:54:00:1d:79:1b   C                     eno1
vlan635.inr-350-reccev.  ether   00:19:a9:c9:88:00   C                     eno1
pestilence.OCF.Berkeley  ether   52:54:00:43:c5:4d   C                     eno1
```

useful commands: arp -e to show entries, arp -d to delete entries, arp -s to add new static entries

data actually in `/proc/net/arp`, `/etc/ethers` for static assignments

```
00:0c:29:c0:94:bf 10.0.0.2
00:0c:59:44:f0:a0 10.0.0.5
. . . .
```

Interface Configuration

`$ ip link` - manage interfaces at L2

`$ ip link set <iface> [up|down]` - enable/disable logical interface

`$ ip link [add|delete] <iface> type [type]` - add/remove interfaces themselves, e.g. bridge or vlan virtual devices

static configuration (on Debian) lives in `/etc/network/interfaces`

/etc/network/interfaces

- auto
  - activate on boot
- iface [name] [family] [type]
  - address family
    - inet -> IPv4
    - inet6 -> IPv6
- config type: static, DHCP
- Additional configuration methods like [pre|post]-up

```
abizer@surge:~$ cat /etc/network/interfaces
auto lo
iface lo inet loopback

auto eno1
iface eno1 inet static
  address 169.229.226.127
  netmask 255.255.255.0
  gateway 169.229.226.1

iface eno1 inet6 static
  address 2607:f140:8801::1:127
  netmask 64
  gateway 2607:f140:8801::1
```

Layer 3: logical network layer

- IP address - 192.168.1.1, 169.229.226.23
- add logical addresses to L2 interfaces
- Can have multiple logical addresses per L2 interface
- Associate logical / L2 interface through bridge device to map multiple MAC/IP pairs into single phys iface
- Use `$ ip addr` to see L3 information

```
abizer@surge:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP gr
    link/ether f8:32:e4:9b:38:ef brd ff:ff:ff:ff:ff:ff
    inet 169.229.226.127/24 brd 169.229.226.255 scope global eno1
       valid_lft forever preferred_lft forever
    inet6 2607:f140:8801::1:127/64 scope global
       valid_lft forever preferred_lft forever
    inet6 fe80::fa32:e4ff:fe9b:38ef/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOW
    link/ether 02:42:0a:f5:96:29 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
4: br0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN g
    link/ether d6:40:f2:77:2a:3d brd ff:ff:ff:ff:ff:ff
    inet6 fe80::d440:f2ff:fe77:2a3d/64 scope link
       valid_lft forever preferred_lft forever
```

IP Addresses

- identify devices connected an Internet Protocol network
- IPv4 (32 bit) and IPv6 (128 bit) addresses
- IPv4 addresses are written in CIDR format, delimited by a dot at each octet (byte)
  - **127.0.0.1** (decimal) / **01111111.00000000.00000000.00000001** (binary)

- Partition block of addresses into networks via masks (format: ip_address/mask)
  - **169.229.226.0/24**
  - mask indicates number of bits to identify network
  - remaining bits identify a host within the network

- Broadcast IP **255.255.255.255** / **11111111.11111111.11111111.11111111**

Managing L3 Addresses

`$ ip addr [iface]` - show all L3 info / on a specific iface

`$ ip addr [add|del] addr/mask dev iface` - add/delete an address to an interface

`$ ip [-6] route` - show routing table

`$ ip [-6] route [add|del|replace] [default] via [address] [dev] dev` - add/remove routes to kernel routing table

Dynamic Host Configuration Protocol

- DHCP - a way for devices to receive IPv4 configuration info from network itself
- IP addresses “leased” from DHCP server to prevent collision, needs to renew
- IPv6 has multiple equivalents: SLAAC, DHCPv6, etc.
- When client requests DHCP, it does an L3/L2 broadcast, and DHCP server returns:
    - An IPv4 address
    - Network Mask
    - Address of first hop (gateway)
    - Possibly, DNS servers for use in local domain

DHCP Tools

`$ dhcpcd <iface>` (IPv4 and IPv6)

`$ dhclient [-4|-6] iface` (deprecated, but still useful in specific cases)

dhcpcd starts a daemon that will attempt to renew leases, so in order to re-dhcp, the daemon needs to be restarted or reloaded

Domain Name System

- DNS: map human-friendly names to IP addresses
- DNS resolver sends DNS query to DNS server to get IP address for name
- Resolution takes place right to left, growing in specificity
- Resolve nyx.ocf.berkeley.edu.
  - 13 root servers hardcoded into every machine to seed request
  - query root server, returns authoritative server address for .edu. zone
  - query .edu zone server, returns address of berkeley.edu zone authority
  - query .berkeley.edu NS, returns address of ocf.berkeley.edu NS (ns.ocf.berkeley.edu)
  - query ns.ocf.berkeley.edu for nyx.ocf.berkeley.edu, get 169.229.226.231

DNS Records

- DNS data stored in form of Resource Records (RR).
- RR are a tuple of (name, value, type, TTL)
- A records - maps hostname to IPv4 address
  - name = hostname
  - value = IP address

```
abizer@surge:~$ dig +nocmd A ocf.berkeley.edu +noquestion +noad
ocf.berkeley.edu.        300      IN       A        169.229.226.23
```

- NS records - authoritative nameserver for zone
  - name = domain
  - value = name of DNS server for domain

```
abizer@surge:~$ dig +nocmd NS ocf.berkeley.edu +noquestion +noaddit
ocf.berkeley.edu.        300     IN      NS      adns2.berkeley.edu.
ocf.berkeley.edu.        300     IN      NS      adns1.berkeley.edu.
```

DNS Tools

$ dig <domain>

$ host <domain>

--

$ rndc reload (reload bind9 zones)

$ nscd -i hosts (flush local DNS cache)

```
tonysitu@death:~$ dig inst.eecs.berkeley.edu

; <<>> DiG 9.9.5-9+deb8u14-Debian <<>> inst.eecs.berkeley.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 60402
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 5, ADDITIONAL: 8

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;inst.eecs.berkeley.edu.                IN      A

;; ANSWER SECTION:
inst.eecs.berkeley.edu. 32963   IN      A       128.32.42.199

;; AUTHORITY SECTION:
eecs.berkeley.edu.      32724   IN      NS      adns1.berkeley.edu.
eecs.berkeley.edu.      32724   IN      NS      ns.CS.berkeley.edu.
eecs.berkeley.edu.      32724   IN      NS      adns2.berkeley.edu.
eecs.berkeley.edu.      32724   IN      NS      ns.eecs.berkeley.edu.
eecs.berkeley.edu.      32724   IN      NS      cgl.UCSF.edu.

;; ADDITIONAL SECTION:
ns.CS.berkeley.edu.     36188   IN      A       169.229.60.61
ns.eecs.berkeley.edu.   37902   IN      A       169.229.60.153
cgl.UCSF.edu.           84948   IN      A       169.230.27.20
adns1.berkeley.edu.     1616    IN      A       128.32.136.3
adns1.berkeley.edu.     10632   IN      AAAA    2607:f140:ffff:fffe::3
adns2.berkeley.edu.     101386  IN      A       128.32.136.14
adns2.berkeley.edu.     8879    IN      AAAA    2607:f140:ffff:fffe::e

;; Query time: 0 msec
;; SERVER: 2607:f140:8801::1:22#53(2607:f140:8801::1:22)
;; WHEN: Sat Oct 07 17:32:40 PDT 2017
;; MSG SIZE  rcvd: 303
```

DNS files

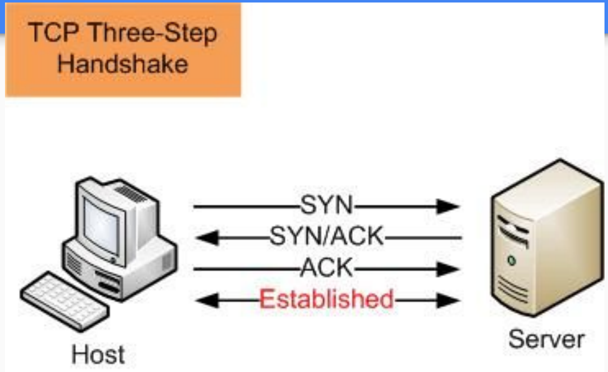
- /etc/hosts
  - statically associate IP addresses with hostnames
  - ip_address canonical_hostname [aliases]
  - 31.13.70.36 www.facebook.com fb ZuccBook myspace.com
  - this is where ‘localhost’ is mapped to 127.0.0.1
- /etc/resolv.conf
  - configures libc DNS resolver
  - one search domain, 3 nameservers, and any number of options
  - nameserver <ip_address>
  - domain <domain_name> vs search <domain_names>

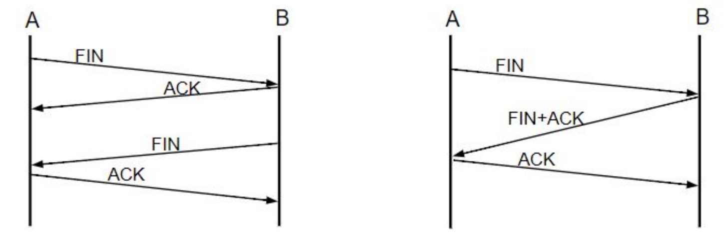
Layer 4 - Transport

- L2 pushes physical bits across the wire, L3 abstracts bits into packets and frames
- L4 abstracts packets and frames into ‘connections’
    - TCP: connection-oriented, reliable, in-order transport
    - UDP: connectionless, unreliable, not in-order transport
- Specify which type of connection when making a socket with SOCK_STREAM (TCP) or SOCK_DGRAM (UDP)

Transmission Control Protocol

- stateful, stream oriented, ensures reliable transport
- mechanisms to guarantee that information arrives intact and in order at the destination
- 4-way handshake to start, 3-way to stop
- Reliability properties create overhead associated with TCP
- Good for usage cases where receiving all data is critical





Two TCP close scenarios with no B-to-A data

User Datagram Protocol

- stateless, connectionless protocol
  - intended for sending messages in datagrams
- no startup/termination overhead
- no guarantees about reliable transport, messages may arrive out of order, or not at all
  - sometimes called Unreliable Datagram Protocol
- many use cases are ok with unreliable transport for low overhead
  - e.g. streaming music and video

Ports and Sockets

- Port identifies a service endpoint on an L3 address
- Socket is an internal endpoint for traffic
  - Associated with a socket address (IP address and port number) and a protocol
- A connection consists of two sockets, one on each host
  - The holy 5-tuple of information:
    (protocol, src_ip, src_port, dst_ip, dst_port)

/proc/net

- Network information available here as virtual files
  - netstat and other tools usually provide a cleaner interface to these

- **/proc/net/dev**
  - Contains information on network devices and statistics like number of bytes received and transmitted

- **/proc/net/[tcp|udp|raw]**
  - Contains information and statistics on open system sockets
  - Used by ss, netstat, etc.

/proc/sys/net

- File interface to internal kernel network configuration
- Edit files:
  - echo [args] > /proc/sys/net/ipv4/ip_forward
  - sysctl -p <conf_file>
  - sysctl -w variable=value
- Subdirectories that can vary from system to system
  - /proc/sys/net/core/
  - /proc/sys/net/ipv4/
- /etc/sysctl.conf to preserve changes
  - net.ipv6.conf.all.disable_ipv6=1

/proc/sys/net/ipv4

- `icmp_echo_ignore_all`
  - Allows kernel to ignore ICMP ECHO packets from every host or only those originating from broadcast and multicast addresses
- `ip_forward`
  - Permits interfaces on the system to forward packets
- `ip_default_ttl`
  - Sets default TTL for outbound packets
- `ip_local_port_range`
  - Specifies range of ports to be used by TCP or UDP when a local port is needed, e.g. ephemeral ports for outgoing connections

Common Tools

host(name) - get DNS info (simple)
ping - see connectivity/latency info
traceroute / mtr - see network path to dest.
arp - view L2/L3 address table
dig / drill - get more detailed DNS info
ip - base of iproute2, complete management of networking subsystems in Linux
netstat / ss - inspect active sockets on system
nc - netcat, simple TCP/UDP client/server

curl / wget - versatile L7 network interaction
iptables - stateful firewall and packet inspection, routing, forwarding (pretty complicated)
ufw - easier-to-use wrapper around iptables
tcpdump - literally dump all packets on an interface

ping

```
tonysitu@death:~$ ping -c 5 google.com
PING google.com (216.58.194.206) 56(84) bytes of data.
64 bytes from sfo03s01-in-f14.1e100.net (216.58.194.206): icmp_seq=1 ttl=55 time=3.20 ms
64 bytes from sfo03s01-in-f14.1e100.net (216.58.194.206): icmp_seq=2 ttl=55 time=3.05 ms
64 bytes from sfo03s01-in-f14.1e100.net (216.58.194.206): icmp_seq=3 ttl=55 time=2.95 ms
64 bytes from sfo03s01-in-f14.1e100.net (216.58.194.206): icmp_seq=4 ttl=55 time=3.15 ms
64 bytes from sfo03s01-in-f14.1e100.net (216.58.194.206): icmp_seq=5 ttl=55 time=2.96 ms

--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 2.957/3.066/3.207/0.106 ms
tonysitu@death:~$ ping -c 5 iceland.is
PING iceland.is (45.76.32.226) 56(84) bytes of data.
64 bytes from 45.76.32.226: icmp_seq=1 ttl=55 time=144 ms
64 bytes from 45.76.32.226: icmp_seq=2 ttl=55 time=144 ms
64 bytes from 45.76.32.226: icmp_seq=3 ttl=55 time=144 ms
64 bytes from 45.76.32.226: icmp_seq=4 ttl=55 time=144 ms
64 bytes from 45.76.32.226: icmp_seq=5 ttl=55 time=144 ms

--- iceland.is ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4004ms
rtt min/avg/max/mdev = 144.082/144.145/144.210/0.243 ms
```

RTT = Round Trip Time

traceroute

Print the route that a packet takes to the destination

```
tonysitu@death:~$ traceroute google.com
traceroute to google.com (216.58.194.206), 30 hops max, 60 byte packets
 1  vlan635.inr-350-reccev.berkeley.edu (169.229.226.1)  1.199 ms  0.958 ms  0.970 ms
 2  t6-6.inr-202-reccev.berkeley.edu (128.32.0.218)  0.623 ms  0.748 ms  0.675 ms
 3  xe-5-2-0.inr-001-sut.berkeley.edu (128.32.0.66)  0.573 ms  0.590 ms  0.579 ms
 4  xe-4-0-0.inr-002-reccev.berkeley.edu (128.32.0.69)  0.578 ms  0.580 ms  0.561 ms
 5  oak-agg4--ucb-10g.cenic.net (137.164.50.30)  2.418 ms  2.323 ms  2.412 ms
 6  74.125.48.172 (74.125.48.172)  3.953 ms  4.368 ms  3.000 ms
 7  108.170.242.225 (108.170.242.225)  2.930 ms  2.923 ms 108.170.243.1 (108.170.243.1)  2.912 ms
 8  108.170.237.105 (108.170.237.105)  2.987 ms 108.170.237.107 (108.170.237.107)  3.196 ms  3.175 ms
 9  sfo03s01-in-f14.1e100.net (216.58.194.206)  2.944 ms  2.996 ms  2.993 ms
```

Details of the number of routers, i.e. 'hops', in the packet path.

How many router hops away is death from supernova? Hint: They are both on the same network (OCF)

0

mtr

combination of ping and traceroute, live, very useful for testing

```
tonysitu@supernova:~$ mtr -rw google.com
Start: Thu Mar  1 15:07:58 2018
HOST: supernova                          Loss%   Snt   Last   Avg  Best  Wrst StDev
  1.|-- vlan635.inr-350-reccev.berkeley.edu  0.0%    10    1.4   1.3   0.8   3.2   0.7
  2.|-- t6-6.inr-201-sut.berkeley.edu        0.0%    10    0.9   2.6   0.8  17.4   5.2
  3.|-- xe-0-2-0.inr-002-reccev.berkeley.edu 0.0%    10    0.8   2.5   0.6  15.3   4.5
  4.|-- 2607:f380:1::118:9a43:21e1          30.0%    10    2.0   2.4   1.6   4.6   0.9
  5.|-- 2001:4860:1:1:0:868:0:28             0.0%    10    2.5   2.6   2.1   3.3   0.0
  6.|-- 2001:4860:0:1004::1                  0.0%    10    3.0   3.1   2.9   3.7   0.0
  7.|-- 2001:4860:0:1::791                  40.0%    10   10.6   5.7   3.5  10.6   3.1
  8.|-- sfo07s26-in-x0e.1e100.net            0.0%    10    2.7   2.7   2.5   2.9   0.0
```

https://linode.com/docs/networking/diagnostics/diagnosing-network-issues-with-mtr/

iproute2

- **ip**
  - Offers a **LOT** of functionality.. You will most commonly be using ip to display/modify routing, IP addresses, or network interfaces.
  - https://access.redhat.com/sites/default/files/attachments/rh_ip_command_cheatsheet_1214_jcs_print.pdf

```
ip address add 192.168.0.77 dev eth0
ip addr del 192.168.0.77/24 dev eth0
```

```
abizer@surge:~$ ip -6 route
2607:f140:8801::/64 dev eno1 proto kernel metric 256  pref medium
fe80::/64 dev eno1 proto kernel metric 256  pref medium
fe80::/64 dev br0 proto kernel metric 256  pref medium
default via 2607:f140:8801::1 dev eno1 metric 1024  pref medium
```

demo:

- **ss -tulpn**
  - socket statistics - tcp,udp,listening,process,numeric
- **tcpdump -i eno1**
  - dump all TCP traffic on eno1
- **nc addr 1234 | nc -l 1234**
  - connect over TCP to port 1234 to a server listening on 1234/tcp

iptables

- A firewall included with most Linux distributions that serves as an interface to setting up traffic filters in the kernel
- Hard to do this topic justice … will defer this to lab
- Additional information can be found at these great links from our sponsors:
    - https://www.digitalocean.com/community/tutorials/how-the-iptables-firewall-works
    - https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewall-using-iptables-on-ubuntu-14-04
    - https://www.digitalocean.com/community/tutorials/iptables-essentials-common-firewall-rules-and-commands
    - https://www.digitalocean.com/community/tutorials/how-to-list-and-delete-iptables-firewal

Questions?

Attendance word: golang

- Basic for head start and reference
    - https://decal.ocf.berkeley.edu/labs/b5